

DESIGN IDEAS

EDITED BY CHARLES H SMALL AND ANNE WATSON SWAGER

8051 converts 16-bit integer to BCD

Jeremy Ottenstein

Allied Signal Aerospace Corp, Teterboro, NJ

For an 8051 μ P, converting a 16-bit binary integer to decimal form is considerably more complicated than converting an 8-bit integer. The most straightforward method uses a 16-bit-divide routine. Listing 1's alternative method takes advantage of the 8051's BCD commands. Using the BCD commands results in simple and clean code.

The listing is for the Boston Systems Office (Waltham, MA) 8051 macro assembler. Its macro defini-

tions differ somewhat from Intel's. The listing uses (r0, r1) and (r2, r3, r4) for the input and output, respectively; but you can use any five 8051 registers or any five internal-direct memory locations. You can obtain the listing from the EDN BBS. Phone (617) 558-4241 with modern settings 300/1200/2400, 8, N, 1. From the main menu, enter (s)ig, <s/di_sig>, rk946).

(EDN BBS /DI_SIG #946)

EDN

To Vote For This Design, Circle No. 746

Listing 1—8051 Radix-conversion routine

```
title   cnvbcd

; Listing controls
$ nocond
$ genonly

rlcl6 macro %x1, %x2
; Performs a 16-bit rotate left through the Carry flag.
    mov     a, %x2
    rlc     a
    mov     %x2, a
    mov     a, %x1
    rlc     a
    mov     %x1, a
endm

clrbcd macro %x1, %x2, %x3
; Initializes the packed bcd digits.
    mov     %x3, #0
    mov     %x2, #0
    mov     %x1, #0
endm

incbcd macro %x1, %x2, %x3
; Increments the packed bcd digits by 1, if the carry is set.
    mov     a, %x3
    addc    a, #0
    da      a
    mov     %x3, a

    mov     a, %x2
    addc    a, #0
    da      a
    mov     %x2, a

    mov     a, %x1
    addc    a, #0
    da      a
    mov     %x1, a
endm

dblbcd macro %x1, %x2, %x3
; Doubles the packed bcd digits.
    mov     a, %x3
    add     a, %x3
    da      a
    mov     %x3, a

    mov     a, %x2
```

```
    addc    a, %x2
    da      a
    mov     %x2, a

    mov     a, %x1
    addc    a, %x1
    da      a
    mov     %x1, a
endm
```

cnvbcd:

; CNVBBCD accepts a 16-bit unsigned binary integer and converts it into
; 5 packed bcd digits.

; The input is passed in (r0,r1).
; The output is returned in (r2,r3,r4).

; Note : The contents of R0 & R1 are destroyed at the end of this routine.

;* The following algorithm is used.
;* See Knuth, "The Art of Computer Programming", Vol 2, 1981
;* Section 4.4, Radix Conversion.

```
;* y := 0 ;
;* for i := 1 to 15 do
;*   y := y + x[16-i] ;
;*   y := y * 2 ;
;* end do ;
;* y := y + x[0] ;
```

```
    mov     r7, #15
    clrbcd  r2, r3, r4
```

```
1$:   rlc16  r0, r1
      incbcd r2, r3, r4
      dblbcd r2, r3, r4
      djnz  r7, 1$
```

```
      rlc16  r0, r1
      incbcd r2, r3, r4
```

ret

end